EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Sayed (reg. 52,544) on 10/22/2009.

The application has been amended as follows:

In the claims:

1. (Currently Amended) A method for analyzing a program, comprising:

providing an application builder that receives source code instructions and determines the minimum amount of the source code instructions that is required by a program;

determining by the application builder a set of functions required by the program by performing a local flow-sensitive type constraint analysis at an intermediate language instruction level to determine which functions have the potential of being executed and to eliminate unused functions in the program, whereby the program is analyzed recursively to determine which virtual functions are called throughout the program; [[and]]

determining a call path for virtual calls and a value graph that may reach a function ~~containing such instruction~~ required by the program; and

analyzing all possible object types that are created at every instruction of the program from the call path by tracking the object types as they pass through plural constructs in the program and calling into functions generically for handling specialized native runtime type information based on the analysis of the object types.

2 (Currently Amended) The method of Claim 1, further comprising:
analyzing a program instruction in the program that accesses an object field, wherein the analysis is performed locally to an object instantiation.

3. (Currently Amended) The method of Claim 1, further comprising: analyzing a program instruction in the program that accesses an array element locally to an array instantiation.

4. (Currently Amended) The method of Claim 1, further comprising:
analyzing a program instruction in the program that accesses the runtime type information for a local runtime symbol usage.

5. (Currently Amended) The method of Claim 1, further comprising:
analyzing a program instruction in the program within an exception handler performed locally to an exception instruction.

7. (Currently Amended) The method of Claim 6, wherein the set of functions ~~may be~~ is in a single program image.

8. (Currently Amended) A [[computer-readable]] computer readable storage medium storing computer-executable process steps of a process for analyzing a program, comprising:

determining, by an application builder that receives source code instructions and

determines the minimum amount of the code instructions that is required by the program, a set of functions required by the program by performing a local flow-sensitive type constraint analysis at an intermediate language instruction level to determine which functions have the potential of being executed and ~~eliminating~~ to eliminate unused functions in the program, whereby the program is analyzed recursively to determine which virtual functions are called throughout the program; [[and]]

determining a call path for virtual calls and a value graph that may reach a function ~~containing such instruction~~ required by the program; and

analyzing all possible object types that are created at every instruction of the program from the call path by tracking the object types as they pass through plural constructs in the program and calling into functions generically for handling specialized native runtime type information based on the analysis of the object types.

9. (Currently Amended) The computer readable storage medium of Claim 8, further comprising:

analyzing a program instruction in the program that accesses an object field, wherein the analysis is performed locally to an object instantiation.

10. (Currently Amended) The computer readable storage medium of Claim 8, further comprising:

analyzing a program instruction in the program that accesses an array element locally to an array instantiation.

11. (Currently Amended) The computer readable storage medium of Claim 8, further comprising:

analyzing a program instruction in the program that accesses the runtime type information for a local runtime symbol usage.

12. (Currently Amended) The computer readable storage medium of Claim 8, further comprising:

analyzing a program instruction in the program within an exception handler performed locally to an exception instruction.

13. (Currently Amended) The computer readable storage medium of Claim 8, further comprising:

declaring possible return types of native functions, where a type analysis of intermediate language instruction is not possible.

14. (Currently Amended) The computer readable storage medium of Claim 13, wherein the set of functions ~~may be~~ is in a single program image.

15. (Currently Amended) A method for analyzing a program, comprising:

determining by an application builder an object type that [[may]] exists at an execution point of the program and evaluating all possible object types that are created at every instruction of [[a]] the program and carrying the object types through a stack evaluation, wherein [[this]] the determination of the object type is performed by a local flow-sensitive type constraint analysis at an intermediate language instruction level and enables determination of a possible virtual function that may be called and allows determination and solutions to program bottlenecks while minimizing virtual functions and dead codes in the program;

creating a call graph at a main entry point of the program;

recording an outgoing function call within a main function of the program;

analyzing possible object types that may occur at any given instruction from any call path for a virtual call in the program, wherein possible object types are determined by tracking the object types as they pass through plural constructs of the program; and

calling into functions generically for handling specialized native runtime type information based on the analysis of the object types.

16-19 (Canceled).

20. (Currently Amended) A [[computer-readable]] computer readable storage medium storing computer-executable process steps of a process for analyzing a program, comprising:

determining by an application builder an object type that [[may]] exists at an execution point of the program enabling determination of a possible virtual function that may be called and evaluating all possible object types that are created at every instruction of [[a]] the program and carrying the object types through a stack evaluation, and providing visibility to functions which are required and also to the chain of dependencies, wherein the determination of the object type is performed by a local flow-sensitive type constraint analysis at an intermediate language instruction level and enables determination of possible virtual functions that may be called;

creating a call graph at a main entry point of the program;

recording an outgoing function call within a main function of the program;

analyzing possible object types that may occur at any given instruction from a call path for virtual calls in the program, wherein the possible object types are determined by tracking the object types as they pass through plural constructs of the program; and

calling into functions generically for handling specialized native runtime type information based on the analysis of the object types.

21-30 (Canceled).

32. (Currently Amended) The computer readable storage medium of Claim 8, wherein the program runs in a managed runtime environment.

34. (Currently Amended) The computer readable storage medium of Claim 20, wherein the program runs in a managed runtime environment.

35-48 (Canceled).

In the specification:

In page 14, [0079], line 19, "Perl" have been changed to – Perl™.

Examiner's Statement of Reason(s) for Allowance

2. Claims 1-15, 20, and 31-34 (renumbered as 1-20) are allowed.

3. The following is an examiner's statement of reason s for allowance:

The prior arts of record, taken alone or in combination, fail to teach or fairly suggest at least:

the program is analyzed recursively to determine which virtual functions are called throughout the program; determining a call path for virtual calls and a value graph that may reach a function required by the program; and analyzing all possible object types that are created at every instruction of the program from the call path by tracking the object types as they

pass through plural constructs in the program and calling into functions generically for handling specialized native runtime type information based on the analysis of the object types recited in claims 1 and 8.

creating a call graph at a main entry point of the program; recording an outgoing function call within a main function of the program; analyzing possible object types that may occur at any given instruction from any call path for a virtual call in the program, wherein possible object types are determined by tracking the object types as they pass through plural constructs of the program; and calling into functions generically for handling specialized native runtime type information based on the analysis of the object types recited in claims 15 and 20.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to INSUN KANG whose telephone number is (571)272-3724. The examiner can normally be reached on M-R 7:30-6 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis A. Bullock, Jr. can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Insun Kang/
Primary Examiner, Art Unit 2193